Claims

What is claimed is:

1. A method for protecting a byte code in a tracing framework, comprising:
   validating a plurality of instructions when loading the byte code; and
   performing at least one safety check while executing the plurality of instructions during a virtual machine emulation,
   wherein the at least one safety check evaluates for a control transfer to an earlier instruction in the byte code sequence.

2. The method of claim 1, further comprising:
   obtaining a tracing program comprising the plurality of instructions.

3. The method of claim 2, further comprising:
   rejecting the tracing program if the byte code is not validated as safe.

4. The method of claim 1, further comprising:
   reporting an error condition and aborting the virtual machine emulation if an unsafe instruction is detected.

5. The method of claim 1, further comprising:
   completing the virtual machine emulation if a safe instruction is detected.

6. The method of claim 1, wherein the transfer comprises one selected from the group consisting of a direct control transfer and indirect control transfer.

7. The method of claim 1, wherein the validating occurs during a single pass over the plurality of instructions.

8. The method of claim 1, wherein the byte code comprises an instruction set of a virtual machine.

9. The method of claim 8, wherein the instruction set comprises at least one selected from the group consisting of an arithmetic operation, a logical operation, a load operation, a store operation, and a control transfer operation.

10. The method of claim 2, wherein the validating the plurality of instructions comprises:

verifying an opcode bits identify a valid operation; and

rejecting the tracing program if the opcode bit is invalid.

11. The method of claim 2, wherein the validating the plurality of instructions comprises:

rejecting the tracing program if an operand name does not refer to a valid operand provided by the virtual machine emulation.

12. The method of claim 2, wherein the validating the plurality of instructions comprises:

computing a destination location within an instruction stream from one of the plurality of instructions; and

rejecting the tracing program if the destination location is invalid.

13. The method of claim 2, wherein the validating the plurality of instructions comprises:

determining whether a subroutine is valid, when one of the plurality of instructions invokes a named subroutine; and

rejecting the tracing program if the subroutine is invalid.

14. The method of claim 2, wherein the validating the plurality of instructions comprises:

summing a total number of the plurality of instructions in the input byte code stream; and

rejecting the tracing program if the total number exceeds a user-configurable limit.

15. The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating whether an arithmetic operation results in a processor exception; and

aborting execution of a tracing program if any exception conditions result.

16. The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the load operation;

determining an appropriate alignment for the effective address; and

aborting execution of the tracing program if the appropriate alignment is improper.

17. The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the load operation before issuing the underlying instructions;

determining an appropriate alignment for the effective address; and

aborting execution of the tracing program if the appropriate alignment is improper.

18. The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the store operation before issuing the underlying instructions;

determining an appropriate alignment for the effective address; and

aborting execution of the tracing program if the appropriate alignment is improper.

19. The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the load or store operation for validity prior to executing the load operation.

20. The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the load or store operation against a list of pre-computed address ranges assigned to a memory-mapped device hardware state; and

aborting execution of the virtual machine emulation if the effective address falls within the list of pre-computed address ranges.

21. The method of claim 9, wherein the performing the at least one safety check comprises:

evaluating an effective address of the store operation against a list of pre-computed address ranges assigned by the virtual machine to the tracing program; and

aborting execution of the virtual machine emulation if the effective address falls within the list of pre-computed address ranges.

22. A mechanism for protecting a byte code, comprising:

an instruction validator configured to validate a plurality of instructions when loading the byte code;

a safety check facility configured to perform at least one safety check while executing the plurality of instructions during a virtual machine emulation,

wherein the at least one safety check evaluates for a transfer to an earlier instruction in the byte code sequence.

23. The mechanism of claim 22, wherein the transfer comprises one selected from the group consisting of a direct control transfer and indirect control transfer.

24. The mechanism of claim 22, wherein the instruction validator is configured to validate during a single pass over the plurality of instructions.

25. The mechanism of claim 22, wherein the byte code comprises an instruction set of a virtual machine.

26. The mechanism of claim 22, wherein the instruction set comprises at least one selected from the group consisting of an arithmetic operation, a logical operation, a load operation, a store operation, and a control transfer operation.

27. The mechanism of claim 22, wherein the instruction validator configured to validate a plurality of instructions comprises:

verifying a opcode bit identifies a valid operation; and

rejecting the tracing program if the opcode bit is invalid.

28. The mechanism of claim 22, wherein the instruction validator configured to validate a plurality of instructions comprises:

rejecting the tracing program if an operand name does not refer to a valid operand provided by the virtual machine emulation.

29. The mechanism of claim 22, wherein the instruction validator configured to validate a plurality of instructions comprises:

computing a destination location within an instruction stream from one of the plurality of instructions; and

rejecting the tracing program if the destination location is invalid.

30. The mechanism of claim 22, wherein the instruction validator configured to validate a plurality of instructions comprises:

determining whether a subroutine is valid, when one of the plurality of instructions invokes a named subroutine; and

rejecting the tracing program if the subroutine is invalid.

31. The mechanism of claim 22, wherein the instruction validator configured to validate a plurality of instructions comprises:

summing a total number of the plurality of instructions in the input byte code stream; and

rejecting the tracing program if the total number exceeds a user-configurable limit.

32. The mechanism of claim 26, wherein a safety check facility configured to perform the at least one safety check comprises:

evaluating whether an arithmetic operation results in a processor exception; and

aborting execution of a tracing program if any exception conditions result.

33. The mechanism of claim 26, wherein a safety check facility configured to perform the at least one safety check comprises:

evaluating an effective address of the load operation;

determining an appropriate alignment for the effective address; and

aborting execution of the tracing program if the appropriate alignment is improper.

34. The mechanism of claim 26, wherein a safety check facility configured to perform the at least one safety check comprises:

evaluating an effective address of the load operation before issuing the underlying instructions;

determining an appropriate alignment for the effective address; and

aborting execution of the tracing program if the appropriate alignment is improper.

35. The mechanism of claim 26, wherein a safety check facility configured to perform the at least one safety check comprises:

evaluating an effective address of the store operation before issuing the underlying instructions;

determining an appropriate alignment for the effective address; and

aborting execution of the tracing program if the appropriate alignment is improper.

36. A computer system for protecting a byte code in a tracing framework, comprising:

a processor;

a memory;

a storage device; and

software instructions stored in the memory for enabling the computer system to:

validate a plurality of instructions when loading the byte code; and

perform at least one safety check while executing the plurality of instructions during a virtual machine emulation,

wherein the at least one safety check evaluates for a control transfer to an earlier instruction in the byte code sequence.

37. The computer system of claim 36, further comprising:

software instructions stored in the memory for enabling the computer system to obtain a tracing program comprising the plurality of instructions.

38. The computer system of claim 36, further comprising:

software instructions stored in the memory for enabling the computer system to reject the tracing program if the byte code is not validated as safe.

39. The computer system of claim 36, further comprising:

software instructions stored in the memory for enabling the computer system to report an error condition and aborting the virtual machine emulation if an unsafe instruction is detected.

40. The computer system of claim 36, further comprising:

software instructions stored in the memory for enabling the computer system to complete the virtual machine emulation if a safe instruction is detected.

41. The computer system of claim 36, wherein the transfer comprises one selected from the group consisting of a direct control transfer and indirect control transfer.